Audio Programming 2

Johan Pauwels

johan.pauwels@uwl.ac.uk

On Today's Programme

Writing audio files

Audio manipulations

Reading audio discussion

- Concept of (dynamic) memory ownership
  - Who is going to clean up memory after usage?
- Avoiding transfers of ownership
  - Easier to keep track
- Knowing when statically allocated memory gets released
- Alternative: using return type for output argument
  - Concept of implicit copies
  - Implications of modern C++11 standard

Memory mgmt guidelines

- Use static allocation when you can, dynamic allocation when you must
- When must you?
  - Array size not known at compile time
  - Keep data after exiting scope (without copy)
  - Stack overflow (stack size: ulimit -s), e.g. data too large, recursion too deep

Extending AudioFile class

- Adding write functionality: new constructor
  - AudioFile(const std::string& file_path, const bool interleaved=true, const bool read=true, const bool write=false);
  - Read and write parameters determine how to open file
  - Interleaved parameter determines how we **interface** with the class (for reading and writing methods, whether data read/written is (non-)interleaved), not whether file stored on disk is interleaved or not
- Actual write functionality:
  - const sf_count_t writeFrames(float* buffer, const int numFrames);
  - Write (append) given (arbitrary) number of frames
  - Can be called multiple times

Audio manipulations

- If we can read audio and write it back to a (different) file
- Then we can manipulate the samples in between => audio effect!
- Simple ones to try
  - Gain (always reduce to avoid clipping)
  - Reversing audio